Word or phrase: instance Search

Find definitions Find translations Search all dictionaries

Descriptions: Verbose, **Compact**

Jump to: General, Art, Business, Computing, Medicine, Miscellaneous, Religion, Science, Slang, Sports, Tech, Phrases

We found 24 dictionaries with English definitions that include the word *instance*:
Tip: Click on the first link on a line below to go directly to a page where "instance" is defined.

General (17 matching dictionaries)

1. instance : Encarta® World English Dictionary, North American Edition [home, info]
2. instance : Merriam-Webster's Online Dictionary, 10th Edition [home, info]
3. instance : Cambridge International Dictionary of English [home, info]
4. instance : The Wordsmyth English Dictionary-Thesaurus [home, info]
5. instance : The American Heritage® Dictionary of the English Language [home, info]
6. instance : Infoplease Dictionary [home, info]
7. instance : Dictionary.com [home, info]
8. instance : UltraLingua English Dictionary [home, info]
9. instance : Cambridge Dictionary of American English [home, info]
10. Instance : Wikipedia, the Free Encyclopedia [home, info]
11. Instance : Online Plain Text English Dictionary [home, info]
12. instance : Webster's Revised Unabridged, 1913 Edition [home, info]
13. instance : Rhymezone [home, info]
14. instance : AllWords.com Multi-Lingual Dictionary [home, info]
15. instance : Webster's 1828 Dictionary [home, info]
16. instance : WordNet 1.7 Vocabulary Helper [home, info]
17. instance : LookWAYup Translating Dictionary/Thesaurus [home, info]

Art (3 matching dictionaries)

18. INSTANCE : Shakespeare Glossary [home, info]
19. instance : The Organon: A Conceptually Indexed Dictionary (by Genus and Differentia) [home, info]

Quick definitions (*Instance*)

***noun*:** an occurrence of something (Example: *"Another instance occurred yesterday"*)
***noun*:** a single item of information that is representative of a type
***verb*:** clarify by giving an example of

Encyclopedia article

In computer science's object-oriented programming, an **instance** is an *instantiated* object of a particular class. (continued)

Do you visit OneLook often? Save yourself some time by dragging the following link into your browser's "Links" toolbar for one-step access to OneLook searches: OneLook

Genus and Differen[redacted]ome, info]
20. instance : Literary C[redacted]ism [home, info]

searches: OneLook

(If you use IE, you may get an alert that says that the link "might not be safe". Don't worry -- it's just because the link uses JavaScript.)

Computing (2 matching dictionaries)

21. instance : Free On-line Dictionary of Computing [home, info]
22. instance : whatis? [home, info]

Miscellaneous (1 matching dictionary)

23. instance : Sound-Alike Words [home, info]

Science (1 matching dictionary)

24. instance : FOLDOP - Free On Line Dictionary Of Philosophy [home, info]

Phrases that include the word ***instance***: for instance, instance variable, court of first instance, formatting output specification instance, causes of instance, *more...*

Words similar to ***instance***: case, example, exemplify, illustrate, illustration, instanced, instancing, representative, *more...*

Additional links for *instance...*

Search completed in 0.366 seconds.

WIKIPEDIA
The Free Encyclopedia

Instance

Main Page
Recent changes
Random page
Current events

Edit this page
Discuss this page
Page history
What links here
Related changes

Special pages
Bug reports

From Wikipedia, the free encyclopedia.

In computer science's object-oriented programming, an **instance** is an *instantiated* object of a particular class.

Edit this page | Discuss this page | Page history | What links here | Related changes

Main Page | About Wikipedia | Recent changes | Go

WIKIPEDIA
The Free Encyclopedia

Object (object-oriented programming)

From Wikipedia, the free encyclopedia.

In computer science, an **object** is something that has an identity, a state, and a behaviour. The state is encoded in instance variables (data members), the behaviour is encoded in methods (member functions). Objects are bundles of related variables and methods and are often used to model real-world objects. Objects can be affected by events. Object-oriented programs typically contain a large number of objects.

It could be said that a class is a blueprint, and an object is a house. An object belonging to a class is referred to as an instance of the class. If humanity were a class, then [you] would be an instance of the class [human].

Overview of Identities, States, and, Behaviours

The identity of an object from the class [dog] might be [Rex]. Its states may include being happy, black, and poodle. Rex can engage in behaviors such as sleeping, barking, and eating. An event affecting Rex might be that he is hit by a car. A program containing an object which represents a bicycle might report such states as velocity or temperature and such behavior as accelerating or the like.

OOP

class 1 class 2

inst/obj. A
inst./obj. B
inst./obj. C

The Atomic Object and Encapsulation

In the atomic or cellular view of an object (or a class), the variables are considered to be within the nucleus and surrounded by methods. In other words, the variables and methods are encapsulated within the object. Each atom (or cell) is modular, that is, modifications to the object do not require encapsulating code to be rewritten, nor do objects need to be located within the same process or computer.

Object-to-Object Communication

Objects can interact and communicate with each other. If object A wants object B to perform one of B's methods, object A will send a message to object B. Consider a program which models driving a

[redacted]e, Object A might be you and Ob[redacted]B might be a car. A message from A-B might involve identifying the object being called upon to perform some action [YourCar], the name of the method (or action) to perform [changeVelocity], and a parameter such as [muchFaster].

Read on: class, object (philosophy), object-oriented programming language, object-oriented programming, object-oriented technology, computing

- List of basic computer science topics

See also: object creation

Object-oriented programming

From Wikipedia, the free encyclopedia.

Object-oriented programming (OOP) is a software design methodology and programming paradigm that defines programs in terms of "classes of objects", objects being entities that combine both *state* (i.e., data) and *behavior* (i.e., procedures, or *methods*).

Table of contents

Definition

The definitions of OOP are vague. In the most general sense, object-oriented programming refers to the practice of viewing software primarily in terms of the "things" (objects) it manipulates, rather than the actions it performs. Other paradigms such as functional and procedural programming focus primarily on the actions, with the objects being secondary considerations; in OOP, the situation is reversed.

Widely-used terminology distinguishes *object-oriented* programming from *object-based*. The former is held to include inheritance (described below), while the latter does not.

Elements

Object-oriented programming expresses a computer program as a collection of objects, which encapsulate data with function and provide regulated communication between each other to perform tasks. This differs from traditional procedural programming in which data and procedures are separate. The objective of object-oriented programming is to componentise software in order to make programs

[redacted]r to write, maintain, reuse and pr[redacted]correct.

Another way this is often expressed is that object-oriented programming encourages the programmer to think of programs primarily in terms of the data types, and secondarily on the operations ("methods") specific to those data types. Procedural languages encourage the programmer to think primarily in terms of procedures, and secondarily the data that those procedures operate on. Procedural programmers write functions, then pass data to them. Object-oriented programmers define objects with data and methods, then send messages to the objects telling them to perform those methods on themselves.

There is some disagreement about exactly which features of a programming method or language qualify as "object-oriented", but there is a general consensus that the following features are most important (each of which is explained in more detail on its own linked page):

- **Abstraction**: Each object in the system serves as a model of an abstract "actor" that can perform work, report on and change its state, and "communicate" with other objects in the system, without revealing *how* these features are implemented. Processes, functions or methods may also be so abstracted, and when they are, a variety of techniques are required to extend an abstraction.
- **Encapsulation**: Also called "information hiding", this ensures that objects cannot change the internal state of other objects in unexpected ways; only the object's own internal methods are allowed to access its state. Each type of object exposes an *interface* to other objects that specifies how other objects may interact with it. Some languages relax this, allowing some direct access to object internals in a controlled way, and limiting the degree of abstraction.
- **Polymorphism**: References to and collections of objects may contain objects of different types, and invoking a behavior on a reference will produce the correct behavior for the actual type of the referent. When it occurs at "run time", this latter feature is called *late binding* or *dynamic binding*. Some languages provide more static ("compile time") means of polymorphism such as C++ templates and operator overloading.
- **Inheritance**: One object's data and/or functionality may be based on those of other objects, from which the former object is said to *inherit*. This allows commonalities among different kinds of objects to be expressed once and reused multiple times. Inheritance is also commonly held to include *subtyping*, whereby

- one type of object is defined to be a more specialised version of another type (see Liskov substitution principle), though non-subtyping inheritance is also possible. Inheritance is typically expressed by describing *classes* of objects arranged in an *inheritance hierarchy* reflecting common behavior.

Object-based programming techniques include the concept of an object (encapsulation, and abstraction) but do not include the object-oriented concepts of inheritance and polymorphism.

History

The concepts of object-oriented programming first took root in Simula 67, a language designed for making simulations, created by Ole-Johan Dahl and Kristen Nygaard of the Norwegian Computing Centre in Oslo. (Reportedly, the story is that they were working on ship simulations, and were confounded by the combinatorial explosion of how the different attributes from different ships could affect one another. The idea occurred to group the different types of ships into different classes of objects, each class of objects being responsible for defining its *own* data and behavior.) They were later refined in Smalltalk, which was developed in Simula at Xerox PARC, but was designed to be a fully dynamic system in which objects could be created and modified "on the fly" rather than having a system based on static programs.

Object-oriented programming "took off" as the dominant programming methodology during the mid-1980s, largely due to the influence of C++, an extension of the C programming language. Its dominance was further cemented by the rising popularity of Graphical user interfaces, for which object-oriented programming is allegedly well-suited. Indeed, the rise of GUIs changed the user focus from the sequential instructions of text-based interfaces to the more dynamic manipulation of tangible components.

Object-oriented features were added to many existing languages during that time, including Ada, BASIC, Lisp, Pascal, and others. Adding these features to languages that were not initially designed for them often led to problems with compatibility and maintainability of code. "Pure" object-oriented languages, on the other hand, lacked features that many programmers had come to depend upon. To bridge this gap, many attempts have been made to create new languages based on object-oriented methods but allowing some procedural features in "safe" ways. Bertrand Meyer's Eiffel was an early and moderately successful language with those goals, but it has now been essentially supplanted by Java, largely due to the emergence of the

[redacted]net, for which Java has special fea[redacted]s. More recently, a number of languages have emerged that are primarily object-oriented yet compatible with procedural methodology, such as Python and Ruby.

Just as procedural programming led to refinements of technique such as structured programming, modern object-oriented software design methods include refinements such as the use of design patterns, design by contract, and modelling languages (such as UML).

Further reading

- Booch, Grady. (1993) ISBN 0805353402 *Object-Oriented Analysis and Design with Applications (Second Edition).* Addison-Wesley.
- Gamma, Erich, Richard Helm, Ralph Johnson, John Vlissides. (1995) ISBN 0201633612 *Design Patterns: Elements of Reusable Object Oriented Software.* Addison-Wesley.
- Meyer, Bertrand. (1997) ISBN 0136291554 *Object-Oriented Software Construction (Second Edition).* Prentice Hall.
- Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen. (1991) ISBN 0136298419 *Object-Oriented Modeling and Design.* Prentice Hall.
- Jacobsen, Ivar. (1994) ISBN 0201544350 *Object-Oriented Software Engineering: A Use Case-Driven Approach.* Addison-Wesley.

External link

- Object-oriented programming FAQ

See also: *Software component, Interface description language, class, object, object-based programming, Object-oriented programming language, Functional programming, Procedural programming, Structured programming, Post-object programming, glossary of object-oriented programming.

Edit this page | Discuss this page | Page history | What links here | Related changes
Other languages: Deutsch | Español | Nederlands | Polski | Français
Main Page | About Wikipedia | Recent changes | Go
Search

Main Page
Recent changes
Random page
Current events

Edit this page
Discuss this page
Page history
What links here
Related changes

Special pages
Bug reports

Class (object-oriented programming)

From Wikipedia, the free encyclopedia.

A **class** is a kind of template (not to be confused with a C++ template, however) that describes the underlying structure of a group of objects. It could be said that a class is a blueprint, and an object is a house. An object belonging to a class is also called an *instance* of that class. If humanity were a class, then [you] would be an instance of the class [humanity].

A class specifies the data items each object of the class contains and the operations or methods that can be performed on each object belonging to the class (in object-oriented languages that lack generic functions).

(Some languages have objects but no classes. In these languages, objects are not restricted to structure provided by classes, and can be changed at will. This less common technique is called object-based programming.)

Subclasses and Superclasses

Classes are often related in some way. The most popular of these relations is inheritance, which involves **subclasses** and **superclasses**, also known respectively as **child classes** (or **derived classes**) and **parent classes** (or **base classes**). If [cars] was a class, then [Jaguar] and [Porsche] might be two sub-classes. If [Button] is a subclass of [Control], then all buttons are controls.

Some programming languages (for example C++) allow multiple inheritance - they allow a child class to have more than one parent class. This technique has been criticized by some for its unnecessary complexity and being hard to implement efficiently, though some projects have undoubtedly benefited from its use. Java, for example has no multiple inheritance, its designers feeling that this would be more trouble than it was worth.

Because the Microsoft Windows API was originally designed in an

object-oriented way, each window is an instance of a class called the "window class".

Sub- and superclasses are considered to exist within a hierarchy.

See also: hierarchy, object-oriented programming, abstract class

For related meanings of the word *"class"*, see Class. For other uses of the word *class* within computer science, see Class (computer science)

- List of basic computer science topics

WIKIPEDIA
The Free Encyclopedia

Instance variable

Main Page
Recent changes
Random page
Current events

Edit this page
Discuss this page
Page history
What links here
Related changes

Special pages
Bug reports

From Wikipedia, the free encyclopedia.

In object-oriented programming, an **instance variable** or **data member** is data which an object keeps track of.

See also: class, class variable, field, instance, method

Edit this page | Discuss this page | Page history | What links here | Related changes

Main Page | About Wikipedia | Recent changes | Go